A题

30%

预处理序列前缀异或和S。暴力枚举子段$[L,R]$，则可$O(1)$询问子段异或和$S_R\ xor\ S_{L-1}$。时间复杂度$O(qn^2)$

60%

预处理序列前缀异或和S及二阶前缀异或和，枚举子段右端点，则需要为答案累加上当前位置的S值若干次，以及S的一个区间异或和。时间复杂度$O(qn)$

100%

首先不考虑区间询问。我们知道有一个经典的问题是询问所有子段和的和，方法是计算每一个元素在总和中出现的次数。这个问题与它非常类似，因为异或与加法具有相似的性质：交换律和结合律。因此要知道总和，只需要知道每一个元素出现的次数即可。

考虑元素a_i，序列$a_1,a_2,\ldots,a_n$中共有$i(n-i+1)$个子段包含它。对于上面的经典问题，答案就是$\sum_{i=1}^{n}a_i i(n-i+1)$。

考虑异或，我们知道，一个数在异或和中出现偶数次相当于没有出现，出现奇数次相当于只出现了一次。观察$i(n-i+1)$，发现n取偶数时i与$n-i+1$奇偶性必然不同，因此乘积为偶数，即所有元素出现的次数都是偶数，答案为0。若n为奇数，则只有i取奇数时$i(n-i+1)$为奇数，答案就是所有奇数位置上元素的异或和。

现在把问题放到区间询问上，那么在区间长度（$R-L+1$）为偶数时答案为0，区间长度为奇数时，答案为区间内所有位置与L奇偶性相同的元素异或和。只需分奇偶预处理前缀异或和，$O(1)$回答区间询问

B题

题目描述的是一个n个点，m条边的无向图。每条边要选择两端之一的点，为它累加上一个1。要求最终被累加了奇数次的点最多。

算法1

看到前20%数据的m非常小，考虑暴力枚举每条边分配给哪边，共有2^m种情况。这里n的范围其实是假的，有用的点最多只有$2m$个。

复杂度视实现不同可能为$O(2^m)$或$O(n2^m)$

算法2

又有20%数据的n非常小，考虑做一个状压dp，状态中记录每一个点当前的奇偶性。具体来说，设$dp[i][S]$表示前i条边分配好之后是否有可能达到S的奇偶性状态。S是一个n位二进制数。转移时考虑当前这条边的两种选择

复杂度：$O(m2^n)$

算法3

对一个连通块，任意取出它的一棵DFS树，那么剩余的边都是返祖边。首先为所有返祖边任意选择一端，然后在DFS树上从下至上考虑每条树枝边：当一个点x的子树内部已经被选择好时，适当地选择x的父边的决策，一定可以使x的奇偶性也为奇数——除了整棵树的根节点。

由此我们发现，适当地选择每条边的决策，一定可以使连通块里至多有一个奇偶性为偶的点。而考虑到奇偶性为奇数的点数与边数同奇偶，这一定是最优的决策。

既然已经知道连通块中奇偶性为奇的点数要么为点数，要么为点数-1，且与边数同奇偶，只需考察点数和边数是否同奇偶，即可知道这个连通块的最大答案。要统计每个连通块的点数和边数，可以采用Flood-Fill或者并查集等方法，复杂度为$O(n+m)$或$O(m+n\alpha(n))$

另一种证法

要证明一个连通块中最多只有一个奇偶性为偶的点，还有另一种更简洁的方法。首先为每条边任意地做出决策。如果改变一条边的决策（将它从一端改为另一端），那么两端的奇偶性都将取反。

通过取反一条两端奇偶性不同的边，可以“移动”一个偶数点。通过取反一条两端都为偶数点的边，可以消除两个偶数点。因此，我们可以将偶数点“运输”到一起，并两两消掉。最后至多剩下一个无法对消的偶数点。

C题

算法1

题目描述的是一个模意义下的0/1背包。

设$f[i][j]$表示在前i个物品中选取，使a_i之和模m为j的最小代价。考虑当前物品是否选取，容易得到：

$$f[i][j]=\min(f[i-1][j],f[i-1][(j-a_i)\bmod m]+c_i)$$

$$f[0][0]=0$$

$$f[0][j]=+\infty,j>0$$

注意式中的模运算应将运算数同余到$[0,m)$中，代码中是`(j-a[i]+m)%m`

对每一个元素，我们希望求出将它去掉后的DP数组。

做一遍DP是$O(nm)$的。枚举每一个元素，将它去掉之后做一遍DP，时间复杂度为$O(n^2m)$，期望得分20分。

算法2

让我们具体地看一下这个DP。按照滚动数组的想法，我们只需维护当前阶段的DP数组，即$f[i]$，一个长度为m的序列。这个序列的每一个位置表示用当前已加入的物品拼出每一个值的最小代价。

我们发现，可以$O(m)$地向DP中加入一个物品（也就是将$f[i-1]$变换为$f[i]$）。而题目要求的，就是对每一个物品，求出将其他所有物品加入DP后的DP数组。

对物品序列中的每一个位置i，预处理出长度为i的前缀中的物品构成的DP数组（记为$pre[i]$），它可以由$pre[i-1]$加入一个物品得到。类似地，预处理后缀DP数组$suf[i]$

那么对一个元素k，只需枚举它左侧和右侧所取的物品a_i之和，即可合并出单单不包含这一物品的DP数组，即$ans[k][s]=\min\{pre[k-1][i]+suf[k+1][j]|(i+j)\bmod m=s\}$

时间复杂度$O(nm^2)$，期望得分50分。

算法3

以$\sqrt{n}$为块长分块，每块以$O(nm)$的时间预处理出包含当前块以外所有元素的DP数组，花费$O(n\sqrt{n}m)$时间

对每个元素，以$O(\sqrt{n}m)$时间从块的预处理结果加入块中的其他元素，得到答案，共花费$O(n\sqrt{n}m)$时间

因此时间复杂度为$O(n\sqrt{n}m)$，期望得分60分。

算法4

考虑在物品序列上分治。分治区间一开始为$[1,n]$，每次划分为两半，分别递归下去。在分治过程中，维护包含当前区间以外元素的DP数组。

对于分治区间$[l,r]$，它被划分为$[l,mid]$和$[mid+1,r]$。以当前DP数组加入$[mid+1,r]$中的所有元素，即可得到$[l,mid]$对应的DP数组，递归下去。右边也同理。

递归到底层，$l=r$时，得到的DP数组是单单不包含当前元素的DP数组，也就是答案。

设$T(n)$为在长度为n的物品序列上分治的时间代价，有$T(n)=\Theta(nm)+2T(\frac{n}{2})$。易知$T(n)=O(nm\log n)$

时间复杂度：$O(nm\log n)$，期望得分100分

一个小提示：对DP数组加入一个物品是本题运行时间的瓶颈，如果希望优化常数的话，这个过程中要避免进行取模、数组复制等操作。但是本题的时限开到了标程的3倍，并且经过验题人的测试，不做这些优化也是能过的。

作者：于剑